

سؤال الإعادة الذاتية:

- تعني السبغ عدد الدالة لنفسها أكثر من مرة
نفس شرط التوقف باستخدام عبارة if الشرطية

تدريب: برنامج يحسب مجموع الأعداد من (1) إلى (n)

```
#include <iostream.h>
void Fact (int x);
void main()
{
    int n, f;
    cin >> n;
    f = Fact(n); cout << f;
}
int Fact (int x)
{
    if (x <= 1)
        return (1);
    else
        return (x + Fact(x-1));
}
```

تدريب: اكتب برنامج باستخدام سؤال الإعادة الذاتية

```
#include <iostream.h>
int pw (int a, int b);
void main()
{
    int x, y, z;
    cin >> x >> y;
```

```

z = pw(x, y); cout << "ln z = " << z;
}

```

```

int pw(int a, int b)
{
    if (b == 0)
        return 1;
    else
        return (a * pw(a, b-1));
}

```

b

$$\begin{aligned}
 z &= pw(2, 3) \\
 &= 2 \times pw(2, 2) \\
 &= 2 \times 2 \times pw(2, 1) \\
 &= 2 \times 2 \times 2 \times pw(2, 0) = 8
 \end{aligned}$$

تقریر: ایک مقامی اسکول، لائسنس یافتہ، جہاں عربی پڑھتے ہیں۔

```

#include <iostream.h>
int mult(int n, int m);
void main()
{
    int x, y, z;
    cin >> x >> y;
    z = mult(x, y); cout << "ln z = " << z;
}
int mult(int n, int m)
{
}

```



```

if (m <= 1)
    return (n);
else
    return (n + mult(n, m-1));
}

```

$$\begin{aligned}
 Z = \text{mult}(3, 4) &= 12 \\
 &= 3 + \text{mult}(3, 3) \\
 &= 3 + 3 + \text{mult}(3, 2) \\
 &= 3 + 3 + 3 + \text{mult}(3, 1) \\
 &= 3 + 3 + 3 + 3 = 12
 \end{aligned}$$

سند المؤشرات:



رقم البايته في الذاكرة يس عنوان.

int x;

مخزن موقع أو عنوان في ذاكرة

هنا x يس 35

المتغير الحقيقي، المتغير، ومقدار الكبر 2 byte

x = 35;

تلك محتوية لعنوان وعلامة القيمة التي

يتم تخزينها داخل عنوان هذا المتغير.

سند القيمة المؤشرات:

نقطة الوصول، لها جزيء عنوانية المؤشرات:

تعريف المؤشرات:

عند التعامل مع المؤشرات نستخدم علامة * (عامل العنوانية غير المباشر).

لغة C

نوع المتغيرات * ب

int * ptr ;

عنوان

ptr : متغير يخزن إلى عنوان (يؤشر إلى) نوعه المتغير (int) (نوع المتغير الذي له الذاكرة يدعى int)

* ptr : قيمة المتغير (محتوى المتغير)

float * ptr ;
char * ptr ;

كتابة المتغيرات

المتغير الذي يخزن إلى عنوان (يؤشر إلى) نوعه المتغير (int) (نوع المتغير الذي له الذاكرة يدعى int)

void main ()

{

int x, * px ;

x = 74 ;

x
74

px = &x : مؤشر يدل إلى عنوان متغير x (نوع المتغير الذي له الذاكرة يدعى int)

* px = * &x ;
= 1

y

z

cout << x ; 74

cout << * px ; 74

العنوان

المتغيرات :

في الواقع يمكن تخزين ذاكرة الحاسب على شكل خريطة مقسم إلى بايتات وكل بايت في ذاكرة له رقم رقم البايته يسمى عنوان في الذاكرة.

لا بد من التفرقة بين رقم البايته ومحتوى العنوان وهو القيمة التي يتم تخزينها داخل عنوان المتغير.

الهيكل المؤشرات

على العاقل يمكن كتابة البرنامج من دون المؤشرات ويمكن كتابته إلا أن استخدام المؤشرات يسهل الوصول المباشر إلى عناوين المؤشرات.

تفريق المؤشرات * وتسمى على القيمة غير المباشرة وهي

اسم المؤشر * النوع المتغيرات

int * ptr ;

مثال :

وهنا لابد من التفرقة بين ptr و *ptr

ptr هي عبارة عن مؤشر يفر إلى int (مؤشر = متغير).

*ptr : تمثل محتوى العنوان وهي القيمة التي يتم تخزينها داخل عنوان المتغير

مثال :

Float * Fptr ;

char * Cptr ;

كما نرى أيضا

استخدام المؤشرات يكتب برنامج جمع عددين

```
#include <iostream.h>
```

```
void main ( )
```

```
{
```

```
int x, y, *px, *py, z;
```

```
x = 10;
```

```
y = 35;
```

```
px = &x; // عنوان x
```

```
py = &y; // عنوان y
```

```
z = *px + *py;
```

نماذج

```
cout << z;
```


}

ملاحظة:

عند استخدام الدالة في البرنامج يمكن للبرامترات أن تكون مؤشرات
عند تعريف الدالة التي بارامترات مؤشرات ~~بها~~ البرامتر *
وأما عند استدعاء الدالة نبقوا بإشارة العنوان (اختارة العنوان قبل
البرامتر:

مثال: $int = sum(int *a, int *b);$
 $c = sum(\&a, \&b);$

تدريب: باستخدام مفهوم الدالة والبرامتر اكتب برنامج يحس عدد بين ايتين:

```
#include <iostream>
int sum (int *a, int *b);
void main()
{
    int x, y, s;
    cin >> x >> y;
    s = sum(&x, &y);
    cout << "sum = " << s << endl;
}

int sum (int *a, int *b)
{
    int c;
    c = *a + *b;
    return(c);
}
```

ملاحظة:

يمكن في الواقع تمرير البرامترات إلى البرنامج الرئيس عن طريق
التمرير البرامترات بالقيمة:

٢- تمرين البرمجة بالأسطر :

٣- تمرين البرمجة بالأسطر :

تمرين ١ : اكتب برنامج يسمح بـ ا ب مجموع عددين معينين عن طريق تمرين البرمجة بالأسطر

```
#include <iostream.h>
```

```
void Sum(int a, int b, int &c);
```

```
void main
```

```
{
```

```
int x, y, z;
```

```
cin >> x >> y;
```

```
Sum(x, y, z);
```

```
cout << "ln S = " << z;
```

```
{
```

```
void Sum(int a, int b, int &c)
```

```
{
```

```
c = a + b;
```

```
}
```

سلسلة البحث : (خوارزمية البحث الثنائي)

٠ ١ ٢ ٣ ٤ ٥ ٦ ٧ ٨ ٩ ١٠ ١١ ١٢ ١٣ ١٤ دليل

٠ ٢ ٤ ٦ ٨ ١٠ ١٢ ١٤ ١٦ ١٨ ٢٠ ٢٢ ٢٤ ٢٦ ٢٨ خاصية متوسطة

١٦ ١٨ ٢٠ ٢٢ ٢٤ ٢٦ ٢٨

$$\frac{0+14}{2} = 7$$

٢٤ ٢٦ ٢٨

٢٤

$$\frac{8+14}{2} = 11$$

$$\frac{12+14}{2} = 13$$

Key = 25 غير موجود

إذا كانت لدينا مجموعة تحتوي على عناصر ليس فيها إذا كانت المجموعة

تحتوي على قيمة تطابق قيمتها مع قيمة البحث عندها تساوي المفتاح Key

عملية ايجاد وسكانه عنام المصفوفة وتسمى عملية البحث

وعلى وجه التحديد البحث التالي يعتمد على حذف نصف عنام المصفوفة بعد كل مقارنة
 وفي المرحلة الاولى نقدر العنصر الواقع في منتصف المصفوفة مع العنصر Key
 اذا طالبك العنصرين يكون العنصر موجود ولا يوجد نصف عنام المصفوفة ولين المصفوفة
 وعند عدم المطابقة نحذف نصف عنام المصفوفة واذا كانت قيمة العنصر
 Key اكبر من النصفه فنحن العنصر على اليسار فالعنصر على اليمين
 لم نكر عملية المقارنة ونحذف نصف عنام المصفوفة عن الحذف عن مصفوفة مكونة
 من عشر واصفوه ولا ونحذف النصف الخالي نقول ان العنصر موجود ونطبع الدليل او نطبع
 العنصر غير موجود

مثال يوضح ايجاد															Key
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	
6	2	4	<u>6</u>	8	10	12	=								
				8	10	12									
				<u>8</u>											
					<u>10</u>	12									
						<u>12</u>									
							8								

موجود :

مثال

```
#include <iostream.h>
int bns(int b[], int sk, int low, int high, int size);
int main() {
```

```
    const int args = 15;
```

```
    int a[args], key, result, i;
```

```
    for (i = 0; i < args; i++) {
```

عنصر المصفوفة ← $a[i] = 2 * i$

```
        cout << "In Key = "; cin >> key;
```

```
    result = bns(a, key, 0, args - 1, args);
```

```
    if (result != -1)
```



```

    cout << "In Found" << result;
    else
    cout << "Not Found";
    return (0);
}

```

حل
البرنامج

```

int bns(int b[], int sk, int low, int high, int size)
{
    int m;
    while (low <= high)
    {
        m = (low + high) / 2;
        if (b[m] == sk)
            return (m);
        else
            if (sk < b[m])
                high = m - 1;
            else
                low = m + 1;
    }
    return (-1);
}

```

النتيجة المطلوبة